

Projet DRONE

Gestion de L'OS Embarqué et de l'environnement Graphique

Pierre-jean TEXIER

Ecole Supérieure des Technologies Electronique Informatique Infographie

13 Février 2014



Sommaire

- 1 Présentation du Projet
- 2 Segment SOL
- 3 Gestion de Projet
- 4 Droit
- 5 Réalisations
- 6 Bilan de LOT
- 7 Conclusion

Présentation du Projet

Projet de FIN d'étude

Présentation du Projet

Projet de FIN d'étude

- “Contexte Industriel”

Présentation du Projet

Projet de FIN d'étude

- “Contexte Industriel”
- PROJET Drone *Next GEN*

Présentation du Projet

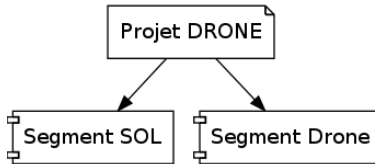
Projet de FIN d'étude

- “Contexte Industriel”
- PROJET Drone *Next GEN*
 - 2 Composants → 2 Equipes

Présentation du Projet

Projet de FIN d'étude

- “Contexte Industriel”
- PROJET Drone *Next GEN*
 - 2 Composants → 2 Equipes



Présentation du Segment SOL

- Présentation



Présentation du Segment SOL

- Présentation



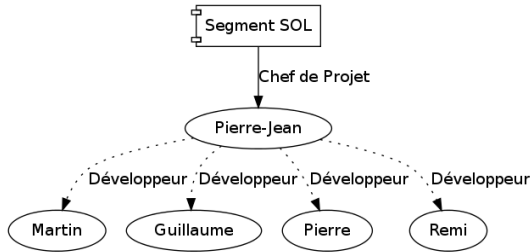
- L'équipe (OBS : Organization Breakdown Structure)

Présentation du Segment SOL

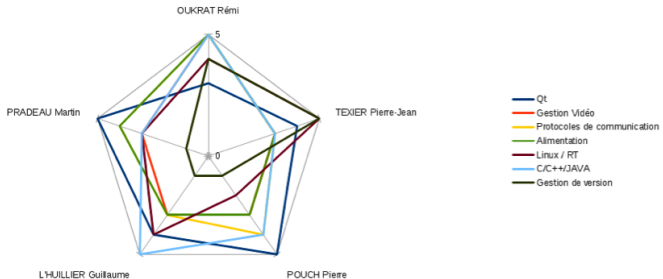
- Présentation



- L'équipe (OBS : Organization Breakdown Structure)



Matrice de compétence Segment SOL



Remarques

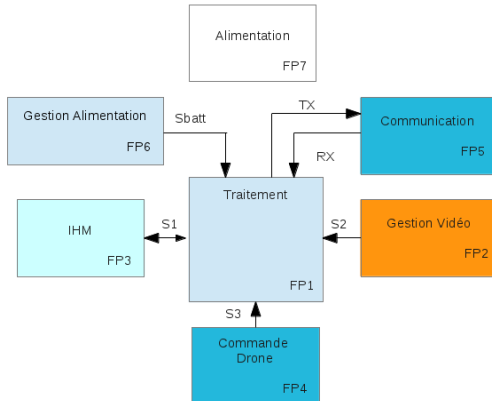
- Permet d'organiser au mieux les ressources

Expression des Besoins

Besoins Exprimés par le Client :

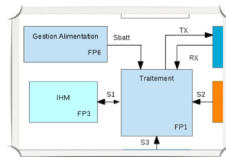
- Affichage
- Ergonomie
- Vidéo
- Communication
- Gamme de Température
 - Commerciale : 0°C à 70°C
 - Industrielle : -45°C à 85°C

Diagramme Fonctionnel de Degré 1



Cahier des Charges Personnel

FP1 - FP6

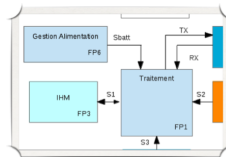


Tâches à réaliser

- OS Linux embarqué Fonctionnel
- Préparation de l'environnement graphique (Qt, openCV, ...)
- Optimisation du temps de boot hardware et subjectif
- Gestion de l'énergie

Cahier des Charges Personnel

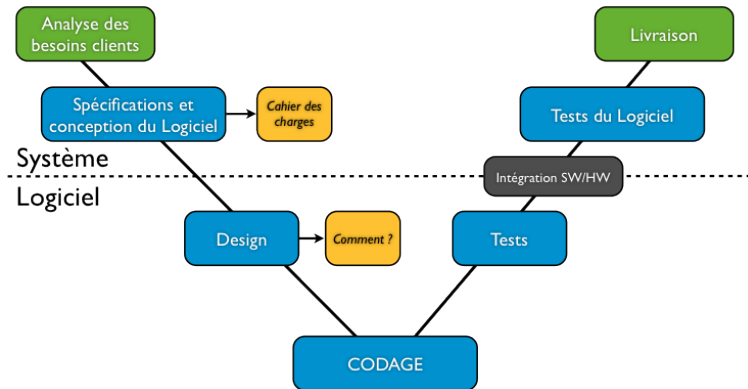
FP1 - FP6



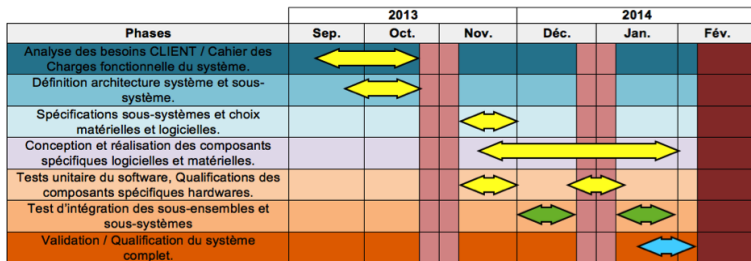
Tâches à réaliser

- OS Linux embarqué Fonctionnel
- Préparation de l'environnement graphique (Qt, openCV, ...)
- Optimisation du temps de boot **hardware** et **subjectif**
- Gestion de l'énergie

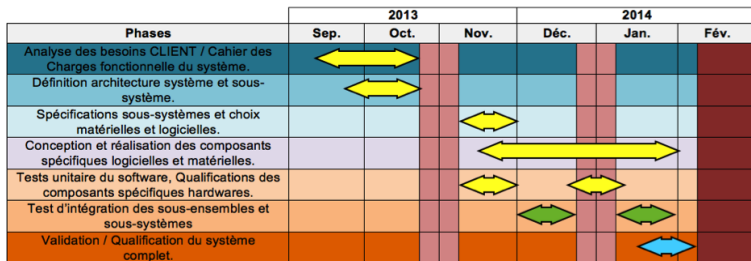
Cycle de vie Logiciel



ROADMAP : Segment SOL



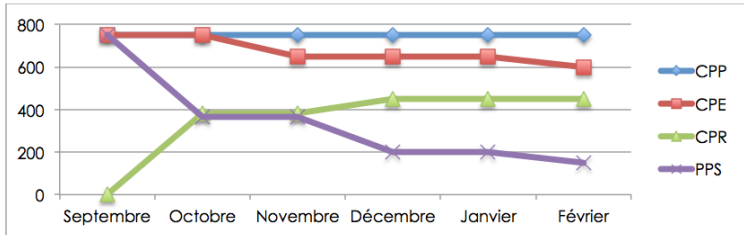
ROADMAP : Segment SOL



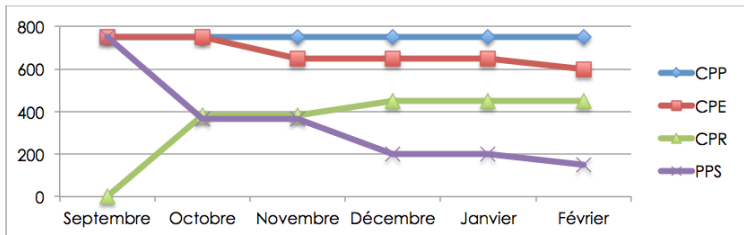
Remarques

- 2 Jalons : Intégration
- Suivi du cycle de Vie Logiciel

Suivi des dépenses : Segment SOL



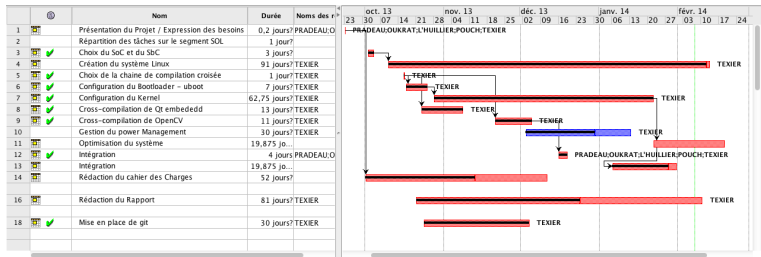
Suivi des dépenses : Segment SOL



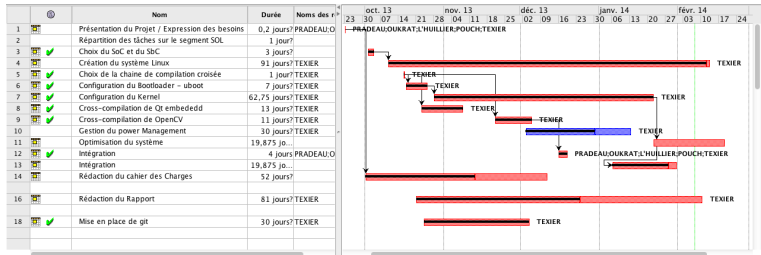
Remarques

- 750 Euros de Budget à l'instant t0
- Environ 450 Euros dépensé en fin de PROJET

Gantt Prévisionnel



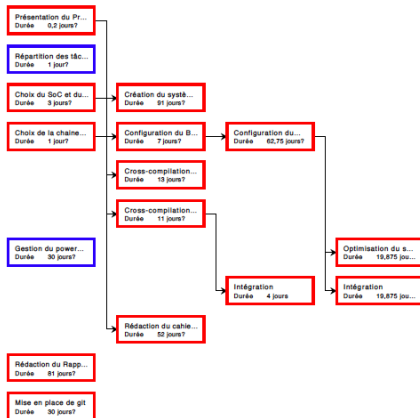
Gantt Prévisionnel



Remarques

- Les tâches du cahier des charges sont présentées ainsi que les relations entre elles

Pert



Outils mis en place

Gestion de Version

- GIT

- ▶ [GITHUB Segment SOL](#)

Gestion de Documentation

- Doxygen

- ▶ [Doxygen Segment SOL](#)

Gestion de Publication

- Doku-Wiki

- ▶ [Wiki Segment SOL](#)

Droit

2 types de Licences utilisés pour le Projet

Droit

2 types de Licences utilisés pour le Projet

- GPLv3 

► Texte GPLv3

Droit

2 types de Licences utilisés pour le Projet

- GPLv3 

► Texte GPLv3

- Creative Commons 

► Texte Creative Commons

Droit

2 types de Licences utilisés pour le Projet

- GPLv3 

► Texte GPLv3

- Creative Commons 

Le plus

- Développement avec des Outils Libre
- Améliore la maintenabilité, portabilité du développement

Droit

2 types de Licences utilisés pour le Projet

- GPLv3 

► Texte GPLv3

- Creative Commons 

► Texte Creative Commons

Le plus

- Développement avec des Outils Libre
- Améliore la maintenabilité, portabilité du développement

Quelques Outils

- GIMP, GNU Linux, GNU plot, bootchart, \LaTeX , fbvis, ...

Réalisations



Choix technologiques

Etude Materiel

- System On Chip



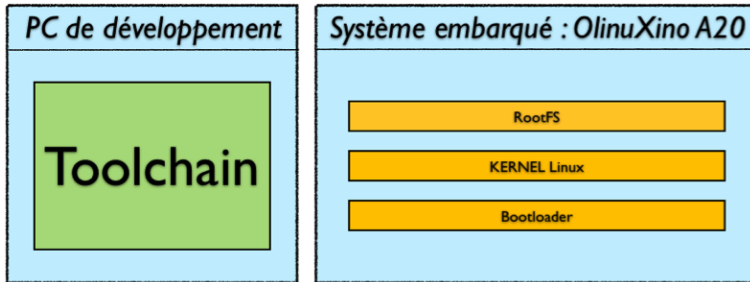
► Synoptique

- Single Board Computer



► Disponible sur le marchés

Environnement “Linux Embarqué”



Chaine de compilation croisée

Compilation Croisée ?


Une chaîne de compilation croisée est un groupe d'outils permettant la compilation d'un programme d'une architecture processeur à une autre (x86 => ARM).

Chaîne de compilation croisée

Compilation Croisée ?

Une chaîne de compilation croisée est un groupe d'outils permettant la compilation d'un programme d'une architecture processeur à une autre (x86 => ARM).

Mon Choix


 -> *arm-linux-gnueabihf-**

Chaine de compilation croisée

Compilation Croisée ?

Une chaîne de compilation croisée est un groupe d'outils permettant la compilation d'un programme d'une architecture processeur à une autre (x86 => ARM).





Mon Choix

 -> *arm-linux-gnueabihf-**

Pourquoi eabihf ?

- HardFloat
- FPU neon-vfpv4

Risques et Opportunités

Composant	Risque			Opportunité		
Bootloader	Créer une image u-boot		Temps	Utiliser l'image présente sur la SD	Temps	
Kernel	Paramétrer/ Recompiler le Kernel	Optimisé pour le projet	Temps	Utiliser l'image présente sur la SD	Temps	Image trop lourde
RootFS	Créer		Temps	Utiliser le rootFS sur la SD	Temps	

Kernel

Kernel ?

► Linux

Kernel

Kernel ?

► Linux

Pourquoi Optimiser ?

Kernel

Kernel ?

► Linux

Pourquoi Optimiser ?

- Empreinte Mémoire

Kernel

Kernel ?

► Linux

Pourquoi Optimiser ?

- Empreinte Mémoire
- Besoins pour le Projet (V4L, Tactile, ...)

Kernel

Kernel ?

► Linux

Pourquoi Optimiser ?

- Empreinte Mémoire
- Besoins pour le Projet (V4L, Tactile, ...)

Informations

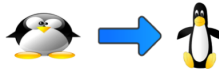
- Version 3.4.67
- Non mainline

► [Lien github](#)

A savoir

- Plusieurs branches
- 3.0 et 3.4.67 = stable
- 3.10 = experimental

Configuration

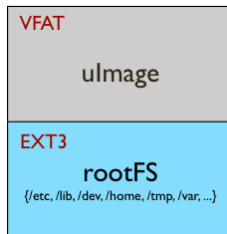


- Interface : xconfig (**make ARCH=arm xconfig**)
- Suppression :
 - Options inutiles dans l'embarqué (ex : Swap)
 - Options de Debug / Profiling (ex : Ftrace)
 - Options Inutiles pour notre Projet (ex : HDMI)

Empreinte Mémoire

Début du Projet : 5.20 MB → Fin du Projet : 3.33 MB

Implantation sur cible




Qt embedded

Fonctionnement : 

Besoins : Génération d'un "qmake" spécifique à notre architecture :

Etapes

- 1 Cross-compilation de la librairie Tactile : tslib 
- 2 Modification du fichier qmake.conf
- 3 Génération du Makefile spécifique aux besoins (./configure)
- 4 Cross-compilation de Qt embedded 4.8.2 (make)
- 5 Installation des binaires (make install)
- 6 Portage des binaires générés sur cible
- 7 Tests

OpenCV embedded



Portage sur Architecture ARM [▶ openCV](#)

Etapes

- 1 Choix des Modules openCV : Utilisation de Cmake
- 2 Cross-compilation librairies/modules
- 3 Portage des binaires générés sur cible
- 4 Tests

Optimisation du temps d'amorçage système



Hardware

U-boot

- Variable 'bootdelay'

Scripts de démarrage : ▶ bootchart

- Networking
- ssh
- exim4
- apache

Subjectif

Remplacement du Logo de démarrage : ***logo_linux_clut224.ppm***

- Logo de base



- Logo personnalisé : 640*480



PBIT

Power On Built in Test

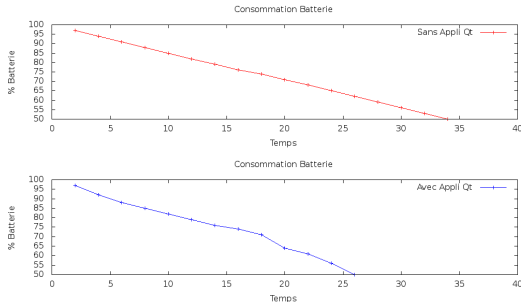
Utilisation simpliste du Framebuffer pour la phase de “PBIT”



Power Management

Réalisé

- Création d'un Crontab
- Test sur l'autonomie du Système



Optimisation du Système

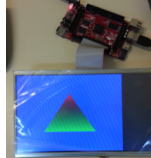
Réalisé

- Système de fichier Temporaire
 - *tmpFS*
- Autologin
 - *agetty –autologin*
- Lancement Automatique de l'application : ihm
- UDEV
 - iDVendor
 - iDProduct

Optimisation du Système

Possible

① Accélération matérielle

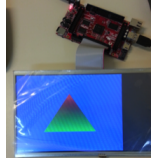


② Gestion des Heuristiques

Optimisation du Système

Possible

① Accélération matérielle



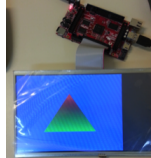
② Gestion des Heuristiques

- powersave

Optimisation du Système

Possible

① Accélération matérielle



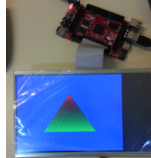
② Gestion des Heuristiques

- powersave
- fantasy

Optimisation du Système

Possible

① Accélération matérielle



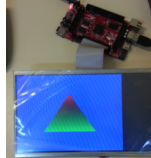
② Gestion des Heuristiques

- powersave
- fantasy
- ondemand

Optimisation du Système

Possible

① Accélération matérielle



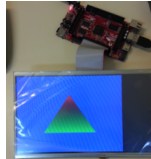
② Gestion des Heuristiques

- powersave
- fantasy
- ondemand
- interactive

Optimisation du Système

Possible

① Accélération matérielle



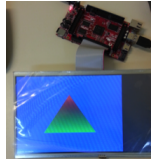
② Gestion des Heuristiques

- powersave
- fantasy
- ondemand
- interactive
- userspace

Optimisation du Système

Possible

① Accélération matérielle



② Gestion des Heuristiques

- powersave
- fantasy
- ondemand
- interactive
- userspace
- performance

LOT Segment SOL : Coûts






Coût de Developpement 

Nom / Prénom	Coût
TEXIER Pierre-jean	3300 €
PRADEAU Martin	2719 €
POUCH Pierre	2640 €
L'HUILLIER Guillaume	2640 €
OUKRAT Rémi	19 669 €

Coût D'industrialisation  : 100 Pièces => 33905 €

Conclusion

Matrice de Validation


Cahier des Charges	TV	Commentaires
Choix SoC / SbC	Levée de risque sur carte	
Chaine de compilation croisée	Compilation "hello world"	
OS Linux sur cible	Sur carte SD	
Préparation graphique	Qt / OpenCV	
Power Management	Via sysFS	

Conclusion

Compétences Acquises

- Portage d'application graphique sur *Architecture ARM*
- Optimisation d'un système Linux
- Gestion d'un Projet de bout en bout : *Chef de Projet*

Bilan Personnel

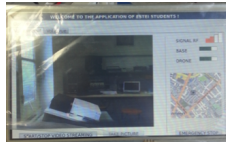
- Mise en pratique de l'enseignement 
- Orientation en *Linux Embarqué* confortée
- Atout pour le prochain stage
- Implication dans la communauté "SUNXI"

FIN

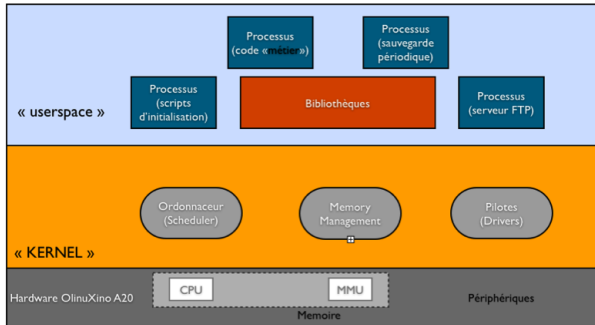
Questions



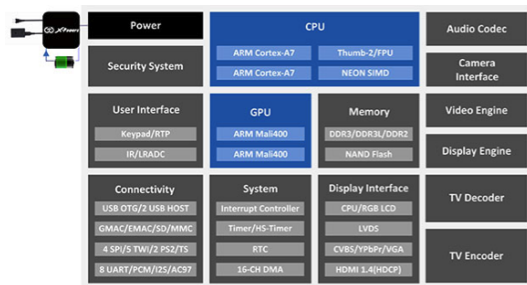
Tests de Validation



Système Linux



Synoptique du System On Chip

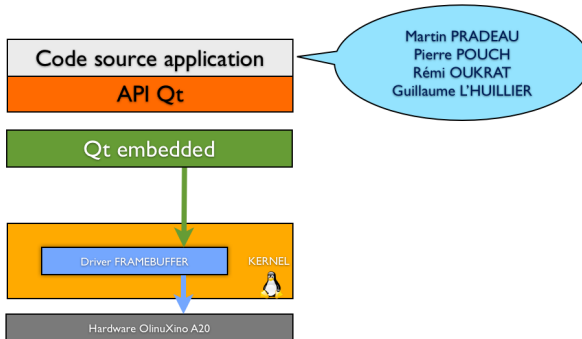


◀ retour



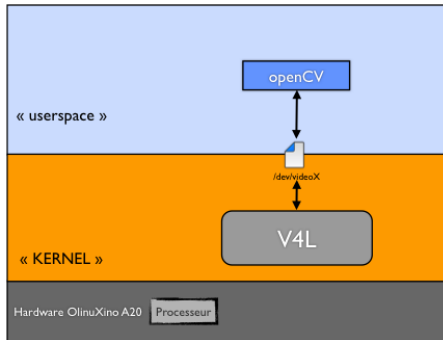
A set of small navigation icons typically found in Beamer presentations, including symbols for back, forward, search, and other slide controls.

Qt embedded



43 / 45

openCV



bootchart

